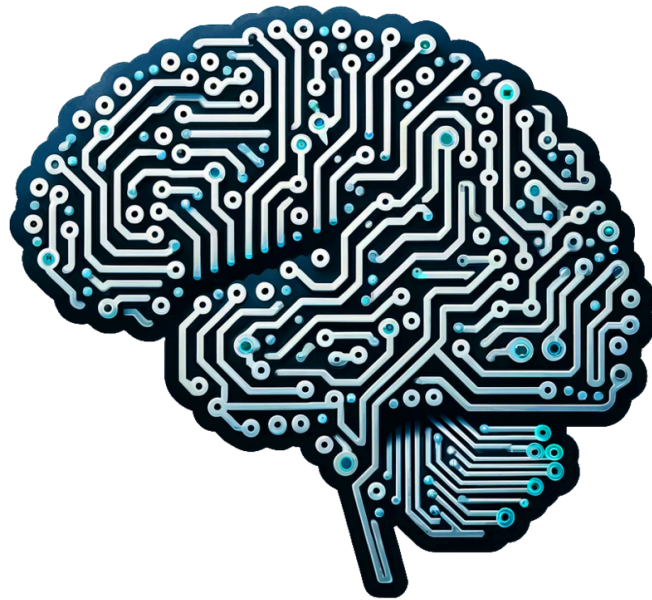


Introducción a ML y Generative AI



Taller de programación #3

Descripción

Aplicar un flujo completo de clasificación utilizando el dataset del Titanic, desde la división de datos hasta la evaluación e interpretación de un modelo de k-NN (k-Nearest Neighbors). Los estudiantes seguirán estos pasos prácticos para entender cómo funciona el modelo y cómo ajustar su rendimiento.

1. Herramientas :

- **Python:** versión 3.11
- **VSCode:** con soporte para Jupyter Notebooks
- **Bibliotecas:** pandas, scikit-learn, matplotlib, seaborn

2. Preparando el Titanic Dataset para clasificación

- Asegúrate de eliminar o imputar valores faltantes.
- Selecciona características relevantes para la predicción (por ejemplo, Pclass, Sex, Age, Fare).
- Convierte las variables categóricas en numéricas (como Sex).
- Define los features y los labels

3. División de datos en entrenamiento y prueba

- Divide los datos en un 80% de entrenamiento y 20% de prueba. Usa una semilla aleatoria fija (por ejemplo, random_state=42) para asegurar que los resultados sean replicables.
- Para esta tarea puedes usar la función train_test_split con el comando (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html):

```
from sklearn.model_selection import train_test_split
```

4. Entrenar el modelo k-NN con datos de entrenamiento

- Entrena un modelo de k-Nearest Neighbors (k-NN) utilizando el conjunto de entrenamiento con un valor inicial de $k=3$.
- Puedes utilizar `KNeighborsClassifier` (<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>)

5. Realizar predicciones en el conjunto de prueba

- Usa el modelo entrenado para predecir las etiquetas del conjunto de prueba.

6. Evaluar el modelo con diferentes valores de k

- Evalúa el modelo con diferentes valores de k (prueba valores como 1, 3, 5, 7, 9).
- Observa cómo el valor de k afecta las métricas de evaluación: Exactitud, Precisión, Recall, y F1-Score.
- Puedes utilizar `accuracy_score`, `precision_score`, `recall_score`, `f1_score` de la librería `scikit-learn` (<https://scikit-learn.org/stable/api/sklearn.metrics.html#classification-metrics>)
- Elige el mejor valor de k .

7. Crea y analiza la Matriz de Confusión utilizando tu mejor k.

- Crea una Matriz de Confusión y explora cómo se distribuyen los errores de clasificación.
- Visualiza la matrix utilizando la librería `seaborn` (<https://seaborn.pydata.org/generated/seaborn.heatmap.html>)
- Interpreta los resultados de la matriz (¿Cuántos falsos positivos y falsos negativos hay?).
- Puedes utilizar `confusion_matrix` de `scikit-learn` (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html)

8. Crea y analiza la Matriz de Confusión utilizando tu mejor k.

- ¿Cuál fue el mejor valor de k?
- ¿Cómo cambió el rendimiento del modelo a medida que ajustaste el valor de k?
- ¿Qué métrica (precisión, recall, F1-score) crees que es más importante para este problema? ¿Por qué?
- Analiza la Matriz de Confusión. ¿Cómo afecta la cantidad de falsos positivos y falsos negativos a la interpretación del modelo?

Marcel Mauricio Moran Calderon
marcel_moran41@hotmail.com

Ariel Ramos Vela
ariel.ramos97@gmail.com